Name:

ID:

CSCI 3104, Algorithms **Profs. Chen & Grochow**

Problem Set 3 – Due Thurs Feb 6 11:55pm **Spring 2020, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solutions:

- All submissions must be easily legible.
- You should submit your work through the **class Canvas page** only.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please allot at least as many pages per problem (or subproblem) as are allotted in this template.

Quicklinks: 1a 1b 2a 2b 3 4

---

1. *Solve the following recurrence relations. For each case, show your work.*

   (a) $T(n) = 2T(n-1) + 1$ *if* $n > 1$*, and* $T(1) = 2$*.*

   We proceed by the unrolling/substitution method. To find the pattern, we will unroll twice. Plugging in $n-1$ and, respectively, $n-2$, we get

$$\begin{aligned} T(n-1) &= 2T(n-2) + 1 \\ T(n-2) &= 2T(n-3) + 1 \end{aligned}$$

   Plugging the first one back into our equation for $T(n)$, we get

$$\begin{aligned} T(n) &= 2T(n-1) + 1 \\ &= 2(2T(n-2) + 1) + 1 \\ &= 4T(n-2) + 3 \end{aligned}$$

   Plugging in our equation for $T(n-2)$ into the above, we get

$$\begin{aligned} T(n) &= 4T(n-2) + 3 \\ &= 4(2T(n-3) + 1) + 3 \\ &= 8T(n-3) + 7 \end{aligned}$$

Now the pattern is fairly clear:

$$T(n) = 2^k T(n-k) + \sum_{i=0}^{k-1} 2^i = 2^k T(n-k) + 2^k - 1 \tag{1}$$

Next we find the k that gets us to the base case, i.e. we solve $n - k = 1$, yielding $k = n - 1$.

Finally, plugging this back into (1), we get

$$\begin{aligned} T(n) &= 2^{n-1}T(1) + 2^{(n-1)} - 1 \\ &= 2^n + \frac{1}{2}2^n = \Theta(2^n). \end{aligned}$$

(b) $T(n) = 3T(\frac{n}{2}) + \Theta(n)$ *if* $n > 1$, *and* $T(1) = \Theta(1)$. *Use the plug-in/substitution/unrolling method.*

We proceed by the unrolling/substitution method. To find the pattern, we will unroll twice. Throughout, we replace the $\Theta(n)$ term by cn, where c is an unknown, positive constant which is an upper bound on the $\Theta(n)$ in the equation for $T(n)$. (Since it is only an upper bound, our equalities all become $\leq$.)

Plugging in $n/2$ and, respectively, $n/4$, we get

$$\begin{aligned} T(n/2) &\leq 3T(n/4) + c(n/2) \\ T(n/4) &\leq 3T(n/8) + c(n/4) \end{aligned}$$

Plugging the first one back into our equation for $T(n)$, we get

$$\begin{aligned} T(n) &\leq 3T(n/2) + cn \\ &= 3(3T(n/4) + c(n/2)) + cn \\ &= 9T(n/4) + cn(3/2 + 1) \end{aligned}$$

Plugging in our equation for $T(n/4)$ into the above, we get

$$\begin{aligned} T(n) &= 9T(n/4) + cn(3/2 + 1) \\ &= 9(3T(n/8) + c(n/4)) + cn(3/2 + 1) \\ &= 27T(n/8) + cn(9/4 + 3/2 + 1) \end{aligned}$$

Now the pattern is fairly clear:

$$T(n) = 3^k T(n/2^k) + cn((3/2)^{k-1} + (3/2)^{k-2} + \cdots + 3/2 + 1) \tag{2}$$

Next we find the k that gets us to the base case, i.e. we solve $n/2^k = 1$, to get $k = \log_2 n$.

Finally, plugging this back into (2), we get

$$T(n) = 3^{\log_2 n} T(1) + cn((3/2)^{\log_2(n)-1} + (3/2)^{\log_2(n)-2} + \cdots + 3/2 + 1)$$

Now we have some work to do to simplify this. First let us rewrite the second part using summation notation (and plug in $c = \Theta(1)$ for $T(1)$, since that was our base case):

$$T(n) = 3^{\log_2 n} c + cn \sum_{i=0}^{\log_2(n)-1} (3/2)^i \tag{3}$$